Exact Sampling with Integer Linear Programs and Random Perturbations

Carolyn Kim
Computer Science Department
Stanford University
ckim@cs.stanford.edu

Ashish Sabharwal
Allen Institute for AI
Seattle, WA
ashishs@allenai.org

Stefano Ermon
Computer Science Department
Stanford University
ermon@cs.stanford.edu

Abstract

We consider the problem of sampling from a discrete probability distribution specified by a graphical model. Exact samples can, in principle, be obtained by computing the mode of the original model perturbed with an exponentially many i.i.d. random variables. We propose a novel algorithm that views this as a combinatorial optimization problem and searches for the extreme state using a standard integer linear programming (ILP) solver, appropriately extended to account for the random perturbation. Our technique, GumbelMIP, leverages linear programming (LP) relaxations to evaluate the quality of samples and prune large portions of the search space, and can thus scale to large tree-width models beyond the reach of current exact inference methods. Further, when the optimization problem is not solved to optimality, our method yields a novel approximate sampling technique. We empirically demonstrate that our approach parallelizes well, our exact sampler scales better than alternative approaches, and our approximate sampler yields better quality samples than a Gibbs sampler and a low-dimensional perturbation method.

Introduction

Probabilistic models are a key component of modern statistical machine learning (ML) and artificial intelligence (AI) systems (Murphy 2012). Inference in such models is a computational challenge that has been intensively researched in physics, statistics, and computer science (Andrieu et al. 2003). The problem is known to be intractable in the worst case, and approximate techniques are often used in practice. One of the most influential notions has been the idea of Monte Carlo approximation, where the computation of an expectation with respect to a complex model is approximated using a sample average. The problem of drawing samples from a general probabilistic model is therefore extremely important, and it is used as a building block inside countless learning, inference, and even numerical analysis algorithms. By far the most popular approach is the Markov Chain Monte Carlo (MCMC) method. The idea is to set up a Markov Chain that, in the limit, will converge to the target distribution, and then draw samples by simulating the chain for a sufficiently long time. Unfortunately, for many models of interest, such chains take exponential time to converge.

We introduce a novel approach for sampling exactly from arbitrary discrete probability distributions. Our approach is based on the Gumbel-max idea (Gumbel and Lieblein 1954), a property of Gumbel random variables with numerous applications in statistics and econometric, and which has recently become popular in the ML community as well (Maddison, Tarlow, and Minka 2014; Gane, Hazan, and Jaakkola 2014; Hazan and Jaakkola 2012; Papandreou and Yuille 2011; Tarlow, Adams, and Zemel 2012; Kappes et al. 2015; Hazan, Maji, and Jaakkola 2013). This property means that one can obtain samples from a probabilistic model by finding the mode of a randomly perturbed distribution obtained by perturbing the log-likelihood of each individual state with i.i.d. Gumbel noise. Modern combinatorial optimization tools such as integer linear programming (ILP) and weighted constraint satisfaction solvers can often handle problems with thousands of variables (Wolsey and Nemhauser 2014) and are well-suited for computing modes of discrete probability distributions. Translating a sampling problem to an optimization problem is therefore a big step forward.

Unfortunately, the Gumbel-max reduction cannot be directly used, as it requires too much "randomness" to generate an exponential number of Gumbel random variables, and too much space to store them (i.e., the resulting optimization problem could not even be encoded compactly). To overcome this issue, we propose an approach inspired by the recent A* sampling algorithm for continuous probability distributions (Maddison, Tarlow, and Minka 2014). While we share with A* sampling the idea of instantiating the randomness as needed in a "lazy" manner, our focus on discrete distributions presents new challenges and opportunities. We show how to make this approach practical using powerful linear programming (LP) relaxations to prune "irrelevant" portions of the state space which can be safely ignored without affecting exactness. We integrate this idea into the commercial ILP solver CPLEX, transforming it into a general-purpose sampler by introducing a Gumbel-based randomization into its branch-and-cut search. Our approach thus directly leverages many recent advances in combinatorial optimization, including parallel search (Boehning, Butler, and Gillett 1988; Johnson, Nemhauser, and Savelsbergh 2000). This opens a novel angle for parallelizing sampling algorithms in a principled fashion.

Our experiments demonstrate the ILP-based sampler

scales to large-treewidth graphical models well beyond the reach of current exact inference methods, including rejection sampling and an adaptation of A* sampling to our discrete setting. Furthermore, even though the approach is general-purpose and oblivious to any special structure in the model, it empirically scales very well on tractable distributions such as tree-structured graphical models, for which LP relaxations employed by CPLEX are known to be tight (Sontag et al. 2008). Our approach also naturally defines a new efficient approximate sampling method, where we simply stop the optimization after a given amount of time, outputting the best solution found so far. Interestingly, information from the LP relaxations (the optimality gap) can be used to evaluate the quality of the approximate sample. This is in stark contrast with traditional MCMC techniques, where there is generally no information on the quality of the results if the chain is stopped earlier, as the chain might have missed important areas of the state space. We demonstrate that our approximate sampler outperforms traditional methods such as a Gibbs sampler as well as a recent low-dimensional perturbation approximation (Hazan, Maji, and Jaakkola 2013).

Preliminaries

Given a finite set Σ and $w : \Sigma \to \mathbb{R}^+$, we seek a randomized algorithm that selects each $\sigma \in \Sigma$ with a probability proportional to $w(\sigma)$, i.e., $p(\sigma) = w(\sigma)/Z$, where Z is the partition function $\sum_{\sigma\in\Sigma} w(\sigma)$. For ease of exposition, we consider $\Sigma = \{0,1\}^n$, but the results extend naturally to general categorical variables.

The Gumbel-max method uses the Gumbel distribution to turn this sampling problem into an optimization problem (Maddison, Tarlow, and Minka 2014). The Gumbel distribution with location k, denoted $\mathrm{Gumbel}(k)$, can be defined by its CDF $\Pr[\mathrm{Gumbel}(k) \le g] = \exp(-\exp(-(g-k)))$. $\mathrm{Gumbel}(k)$ has mean $k + C_E$, where $C_E \approx 0.5772$ is the Euler-Mascheroni constant, and variance $\pi^2/6$. Perhaps more intuitively, we can view $\mathrm{Gumbel}(k)$ as approximating $\max_{1\le n\le \exp(k)} Y_n$, where Y_n are i.i.d. standard exponential distributions; and in fact, as k approaches infinity, $\max_{1\le n\le\exp(k)} Y_n - k$ converges pointwise to $\mathrm{Gumbel}(0)$ (Leadbetter, Lindgren, and Rootzén 1983). For $b \in \mathbb{R}$, the Gumbel distribution "truncated" at b, i.e., forced to be at most b, is defined by its CDF $\Pr[\mathrm{TruncGumbel}(k,b) \le g] = \exp(-\exp(-(\min(g,b)-k)))/\exp(-\exp(-(b-k)))$

The relevant properties of the Gumbel distribution are as follows (Maddison, Tarlow, and Minka 2014). Let $\{\gamma_\sigma \mid \sigma \in \Sigma\}$ be i.i.d. $\mathrm{Gumbel}(0)$ random variables. Then:

$$\max_\sigma\{\gamma_\sigma + \log(w(\sigma))\} \sim \mathrm{Gumbel}(\log Z) \tag{1}$$

$$\mathbb{E}\left[\max_\sigma\{\gamma_\sigma + \log(w(\sigma))\}\right] = \log Z + C_E \tag{2}$$

$$\Pr\left[\sigma = \arg\max_\sigma\{\gamma_\sigma + \log(w(\sigma))\}\right] = \frac{w(\sigma)}{Z} \tag{3}$$

These properties provide a way of computing the partition function (eq. 2) and sampling from the distribution (eq. 3).

MIP Gumbel sampling

Our algorithm is closely related to A* sampling, a recently introduced algorithm that exploits the Gumbel-max method to draw (exact) samples from continuous density functions (Maddison, Tarlow, and Minka 2014). Like A* sampling, we also generate a sequence of i.i.d. Gumbel distributed random variables in a "lazy" and top-down manner. Specifically, the (exponentially) large number of Gumbel variates needed to fully specify the optimization problems in (2) and (3) is only instantiated as needed by a branch-and-bound search procedure. The crucial observation is that in order to decide whether a node v can be pruned, it is sufficient to know the maximum value of the Gumbel random variables at the leaves of the subtree rooted at v, i.e., $\max\{\gamma_\sigma \mid v \text{ is an ancestor of } \sigma\}$. Thanks to a property of Gumbel distributions (eq. 1), this value can be easily sampled without actually instantiating all the other leaves.

Our approach is however tailored for discrete domains, which presents several opportunities and advantages. We can employ techniques from combinatorial optimization such as tractable relaxations to obtain bounds on the objective, which can be used in a branch-and-bound scheme to prune large portions of the search space. These approaches are usually much more effective than their continuous counterparts (Schlesinger 2009). In particular, we use LP relaxations, which are known the be very effective in dealing with sparse structure of real world graphical models. Further, we can leverage decades of engineering in ILP solvers.

ILP formulation

We consider an ILP formulation for the MAP inference problem $\max_\sigma w(\sigma)$. For simplicity of exposition, we focus on binary factors (i.e., pairwise interactions between variables), where $w(\sigma) = \prod_{i\in V}\psi_i(\sigma_i)\prod_{(i,j)\in E}\psi_{ij}(\sigma_i,\sigma_j)$ for some edge set E. Rewriting in terms of log-potentials, the MAP inference problem can be stated as $\max_{\sigma\in\Sigma}\sum_{i\in V}\theta_i(\sigma_i) + \sum_{(i,j)\in E}\theta_{ij}(\sigma_i,\sigma_j)$. This, in turn, may be written as an ILP Q using binary variables $\{\mu_i \in \{0,1\} \mid i \in V\}$ and $\{\mu_{ij}(c_i,c_j) \in \{0,1\} \mid (i,j) \in E, c_i, c_j \in \{0,1\}\}$ (Wainwright and Jordan 2008):

$$\max_{\mu_i,\mu_{ij}} \sum_{i\in V}\left(\theta_i(1)\,\mu_i + \theta_i(0)\,(1-\mu_i)\right) + \sum_{(i,j)\in E}\sum_{c_i,c_j}\theta_{ij}(c_i,c_j)\,\mu_{ij}(c_i,c_j) \tag{4}$$

subject to the following constraints for all $i \in V$, $(i,j) \in E$: $\sum_{c_j\in\{0,1\}}\mu_{i,j}(0,c_j) = 1-\mu_i$; $\sum_{c_j\in\{0,1\}}\mu_{i,j}(1,c_j) = \mu_i$; $\sum_{c_i\in\{0,1\}}\mu_{i,j}(c_i,0) = 1-\mu_j$; $\sum_{c_i\in\{0,1\}}\mu_{i,j}(c_i,1) = \mu_j$.

The objective function of Q can be upper bounded by solving its LP relaxation $LPrelax(Q)$, obtained by allowing binary variables to take values in the continuous range $[0,1]$ and turning the above constraints into linear inequalities.

Gumbel Sampling Using ILP

Algorithm 1 describes our sampling method, called GumbelMIP, that takes as input an ILP Q. It keeps track of the

Algorithm 1 GumbelMIP(Q: ILP with n variables)

1: $X_Q \leftarrow$ variables of Q ▷ $|X_Q| = n$
2: $u \sim \mathit{Uniform}[0,1]$; $\gamma_Q \leftarrow n \log 2 - \log(-\log(u))$ ▷ sample $\gamma_Q \sim \mathit{Gumbel}(n \log 2)$
3: $\sigma \sim \mathit{Uniform}(\{0,1\}^n)$ ▷ uniform random configuration associated with γ_Q
4: $\sigma^* \leftarrow \sigma$; $v^* \leftarrow v(\sigma) + \gamma_Q$ ▷ initialize current best configuration
5: $L \leftarrow \{(Q, X_Q, \gamma_Q, \sigma)\}$ ▷ the set of currently open sub-problems
6: **while** L is non-empty **do**
7: Select and remove $(P, X_P, \gamma_P, \sigma)$ from L ▷ according to sub-problem selection heuristic
8: Solve the LP relaxation of sub-problem P
9: **if** LP relaxation is infeasible **then** Continue **end if**
10: Otherwise denote the LP solution by τ with objective value v
11: **if** $v + \gamma_P \leq v^*$ **then** Continue **end if** ▷ the subtree is suboptimal and can be pruned
12: **if** X_P is empty **then** ▷ τ must equal σ and thus be integer valued
13: $\sigma^* \leftarrow \sigma$; $v^* \leftarrow v + \gamma_P$
14: Continue
15: **end if**
16: Select x_ℓ from X_P ▷ according to branching heuristic
17: Branch on x_ℓ and create 2 new problems $P[x_\ell = 0]$ and $P[x_\ell = 1]$
18: $u \sim \mathit{Uniform}[0,1]$ ▷ sample $\tilde{\gamma} \sim \mathit{TruncGumbel}((|X_P| - 1)\log 2, \gamma_P)$
19: $\tilde{\gamma} \leftarrow (|X_P| - 1)\log 2 - \log\left(\exp(-\gamma_P + (|X_P| - 1)\log 2) - \log u\right)$
20: Let $\tilde{\sigma}$ be σ with $\tilde{\sigma}_\ell = 1 - \sigma_\ell$ and values for $X_P \setminus \{x_\ell\}$ resampled uniformly from $\{0,1\}^{|X_P|-1}$
21: **if** $v(\tilde{\sigma}) + \tilde{\gamma} > v^*$ **then** $\sigma^* \leftarrow \tilde{\sigma}$; $v^* \leftarrow v(\tilde{\sigma}) + \tilde{\gamma}$ **end if**
22: Add $(P[x_\ell = \sigma_\ell], X_P \setminus \{x_\ell\}, \gamma_P, \sigma)$ and $(P[x_\ell = 1 - \sigma_\ell], X \setminus \{x_\ell\}, \tilde{\gamma}, \tilde{\sigma})$ to L
23: **end while**
24: return σ^*

current best configuration σ^* and its Gumbel-perturbed objective value v^*. It also maintains a set L of currently open sub-problems, each defined by an ILP P obtained by freezing the value of a subset of the variables of Q. The sub-problem P is associated with its set of free variables X_P, the max γ_P of (implicit) Gumbel perturbations of all configurations consistent with P (i.e., those in the feasible set of P), and a special configuration σ consistent with P whose Gumbel perturbation γ_σ is fixed to be γ_P. L initially contains Q, along with the max of 2^n independent Gumbel samples and a uniformly randomly chosen configuration σ. As noted earlier, the max of 2^n independent Gumbel samples is itself Gumbel distributed, and is thus easy to sample (Line 2). σ becomes the current best configuration σ^* and its log-weight (the ILP objective value for σ) perturbed by γ_Q becomes v^*.

The branch-and-bound search for the ILP proceeds as follows. Using a sub-problem selection heuristic (e.g., the default in CPLEX), we choose $(P, X_P, \gamma_P, \sigma)$ from L. If the LP relaxation of P is infeasible or if the LP relaxation objective perturbed by γ_P isn't an improvement over the current best perturbed objective v^*, we simply discard P. Otherwise, if P has no free variables, we must have an integer feasible solution in σ tied to the Gumbel perturbation γ_P. We use this to update our running bound v^*, if it's an improvement. Then P is discarded. If P does have free variables, we choose one, say x_ℓ, using a heuristic (e.g., the default in CPLEX, but over-ridden by a forced variable selection step in case all variables already have an integer value in the LP relaxation, which would normally obviate the need for an ILP solver to branch any further; a feasible integer solution might however not be tied to any Gumbel perturbation).

The branching step now creates two new sub-problems, by fixing x_ℓ to 0 and 1, resp. The sub-problem with $x_\ell = \sigma_\ell$ inherits the perturbation γ_Q along with the configuration σ tied to it. The sub-problem with $x_\ell = 1 - \sigma_\ell$ gets a freshly sampled truncated Gumbel perturbation $\tilde{\gamma}$ that is distributed as the max of $2^{|X|-1}$ independent Gumbels each truncated to be no larger than γ_P (Line 19). In this sub-problem, $\tilde{\gamma}$ is tied to a uniformly randomly chosen configuration $\tilde{\sigma}$ that is consistent with σ on all frozen variables of P. If $\tilde{\sigma}$ improves upon the current best σ^*, the latter is updated. After creating the two sub-problems, P is discarded.

The branch-and-bound search continues as above until L becomes empty, at which point it outputs σ^* as the sample and v^* as its Gumbel perturbed objective value.

Relationship with A* sampling

Our method takes advantage of abstractions and algorithms tailored to discrete spaces, resulting in a few technical differences from the A* sampling approach for continuous spaces.

Use of LP relaxations: LP relaxations have been shown to be extremely effective for MAP/MPE inference and combinatorial optimization (Wolsey and Nemhauser 2014; Sontag et al. 2008). Our work demonstrates that LP relaxations can also speed up exact sampling.

Inheritance of configurations: The branching employed by A* sampling would partition a search space into 3 parts: a singleton set $\{\tilde{\gamma}\}$ and two other disjoint subspaces. The standard decomposition of discrete spaces into subtrees (correspoding to branching on a variable) is, however, much more natural and amenable to compact representations, as well as to the use of LP relaxations. At each branching point, rather

than sampling two fresh truncated Gumbel perturbations for the children, we instead have one child inherit the perturbation of the parent (Line 22). This preserves correcteness and allows for traditional branching based on subtrees.

Heuristics for branching: Branching heuristics play a major role in combinatorial search. A* sampling always selects the subproblem with the highest upper bound (best first). While this is a natural heuristic, other more advanced heuristics are used in modern combinatorial optimization. Building on the commercial solver CPLEX, GumbelMIP can leverage decades of research on branching heuristics.

Parallelized searches: Similarly, significant progress has been made in the past 20 years on parallelizing combinatorial search. Our approach can take advantage of these techniques, dramatically reducing the runtime (see Figure 1a).

Analysis

The soundness of search space pruning based on LP relaxations guarantees that upon termination, σ^* output by GumbelMIP will be the optimal solution of the perturbed optimization problem. This, together with Gumbel-max properties discussed earlier and prior observations about lazy instantiation of randomness (Maddison, Tarlow, and Minka 2014), thus guarantee the correctness of GumbelMIP.

The runtime is worst-case exponential, because Algorithm 1 might have to explore the entire (exponentially large) search space. This is consistent with the hardness of sampling, which is believed to be intractable in the worst-case (Jerrum and Sinclair 1997; Koller and Friedman 2009). The empirical evaluation of the runtime is discussed in the experimental section. Intuitively, certain "easy" classes of probability distributions can be handled efficiently. For example, consider the case $w(\sigma) = 1$ for all σ, i.e., a uniform probability distribution over Σ. Then it can be shown that the first configuration σ sampled in line 4 of Algorithm 1 will be the optimal solution for the perturbed optimization problem. Assuming the LP relaxation provides a tight upper bound, Algorithm 1 will be able to prune all nodes and finish the search at the root node.

What happens if the search is aborted before the ILP is solved to optimality? We show that the approach is robust in the sense that its performance degrades gracefully with optimality gap (i.e., how far we are from the optimum).

To gain some intuition, suppose we abort the search when there are still open sub-problems in the set L that might potentially improve upon the current best solution σ^*. Let $\overline{S} = \cup_{(Q,X_Q,\gamma_Q,\sigma)\in L} \mathit{Feasible}(Q) \setminus \{\sigma\}$ where $\mathit{Feasible}(Q)$ denotes the set feasible (integer) solutions of the ILP Q. Let $S = \{0,1\}^n \setminus \overline{S}$ denote its complement, i.e., the space that has actually been explored so far. By design, GumbelMIP maintains the invariant $\sigma^* = \arg\max_{\sigma\in S} \log w(\sigma) + \gamma_\sigma$, i.e., σ^* is the best solution found so far. For any set S, we know from Gumbel-max properties that $\Pr_\gamma[\sigma' = \arg\max_{\sigma\in S} \log w(\sigma) + \gamma_\sigma] = \frac{w(\sigma')}{\sum_{\sigma\in S} w(\sigma)} = p(\sigma' \mid \sigma \in S)$. As expected, the larger the set S is, the better the samples are. When $S = \Sigma$, i.e., we have solved the problem to optimality, we are sampling from the desired target density $p(\sigma)$.

Another way to measure optimization progress is to consider the *rank* of the current-best solution σ^*. Suppose σ^* has the k-th largest perturbed objective value. Intuitively, the smaller k is, the closer we are to the optimal solution and can expect better samples. This intuition is formalized by the following result,[1] which generalizes Gumbel-max properties and relates the rank of the top k solutions (according to the Gumbel-perturbed objective) to sampling without replacement from the original probability distribution $p(\sigma)$:

Theorem 1. *Let $\gamma \in \mathbb{R}^\Sigma$ be a vector of $|\Sigma|$ samples drawn i.i.d. from* $\mathrm{Gumbel}(0)$. *Let $w : \Sigma \to \mathbb{R}^+$ and $w'(\sigma) = \log w(\sigma) + \gamma_\sigma$. Then,*

$$\Pr\Big[w'(\sigma^{(1)}) \geq \cdots \geq w'(\sigma^{(k)}) \geq w'(\sigma^{(\ell)})\ \forall \ell > k\Big] = \frac{w(\sigma^{(1)})}{Z} \frac{w(\sigma^{(2)})}{Z - w(\sigma^{(1)})} \cdots \frac{w(\sigma^{(k)})}{Z - \sum_{j=1}^{k-1} w(\sigma^{(j)})}$$

Interestingly, the information provided by the open nodes in the set L and their LP relaxations allows us to (probabilistically) estimate the rank of the current best solution σ^*.

Proposition 1. *Suppose GumbelMIP stops early, outputs σ^*, and has a set L of open sub-problems. Let k be the rank of σ^* when all configurations are ordered by decreasing $w'(\sigma) = \log w(\sigma) + \gamma_\sigma$. Then,*

$$\mathbb{E}_\gamma[k] \leq 1 + \sum_{\substack{(Q,X,\gamma_Q,\sigma)\in L \\ \mathit{LPrelax}(Q)+\gamma_Q \geq w'(\sigma^*)}} 1 + \Big(2^{|X|} - 1\Big) \cdot \Big(1 - F\big(w'(\sigma^*) - \mathit{LPrelax}(Q), \gamma_Q\big)\Big)$$

where $F(g, b)$ is the CDF of $\mathrm{TruncGumbel}(0, b)$ *evaluated at g.*

LP relaxations of the open nodes in L (and, implicitly, the optimality gap) not only provide us with a way to evaluate the quality of the approximate samples produced by early-stopping (through Theorem 1 and Proposition 1) but can also be used to provide any-time upper and lower stochastic bounds on the value of the partition function Z:

Theorem 2. *Suppose GumbelMIP stops early. Let $v^* = w'(\sigma^*)$ and $U = \max_{(Q,\cdot,\gamma_Q,\cdot)\in L} \mathit{LPrelax}(Q) + \gamma_Q$. Let $C_E \approx 0.5772$ be the Euler-Mascheroni constant. Then,*

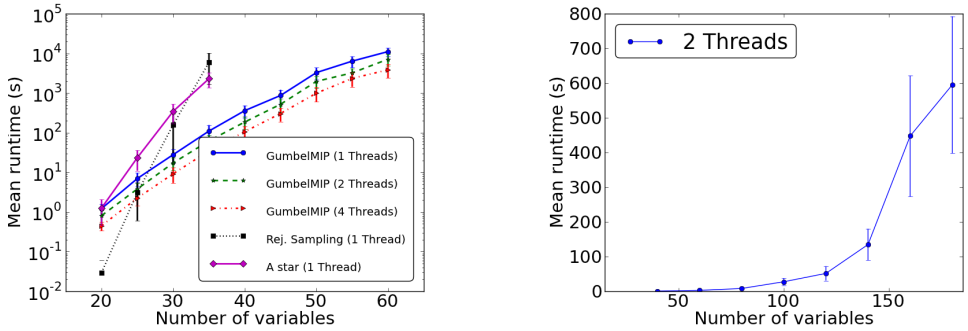
$$\mathbb{E}_\gamma[v^*] \leq \log Z + C_E \leq \mathbb{E}_\gamma[U].$$

Further, for any $\epsilon > 0$ and $\delta > 0$, suppose we repeat the algorithm $T \geq (1/\delta - 1)\pi^2/(6\epsilon^2)$ times using independently generated Gumbel perturbations $\{\gamma^{(t)}\}_t$. Let $\{v_t^*\}_t$ and $\{U_t\}_t$ be the corresponding samples of v^* and U, resp., obtained by stopping the algorithm early. Then,

$$\Pr\left[\log Z + C_E \geq \left(\frac{1}{T}\sum_{t=1}^{T} v_t^*\right) - \epsilon\right] \geq 1 - \delta$$

$$\Pr\left[\log Z + C_E \leq \left(\frac{1}{T}\sum_{t=1}^{T} U_t\right) + \epsilon\right] \geq 1 - \delta$$

[1] Due to limited space, all proofs are deferred to the companion technical report (Kim, Sabharwal, and Ermon 2015).



(a) Attractive cliques.

(b) Tree structured.

Figure 1: Average runtime of exact sampling (over 100 samples) for hard and easy graphical models.

While sampling-based lower bounds on Z can typically be obtained using importance sampling and its variants (Koller and Friedman 2009; Gogate and Dechter 2007; 2011), upper bounds are much more difficult to obtain, requiring either an exponential number of samples or strong conditions on the skew of $w(\sigma)$ (cf. Theorem 12.1 of Koller and Friedman (2009), Chakraborty et al. (2014)). In contrast, Theorem 2 requires a polynomial number of samples and does not make any assumption on $w(\sigma)$. The bounds, of course, may not be tight if GumbelMIP is stopped too early, but they are guaranteed to be correct.

Experimental evaluation

We implemented GumbelMIP on top of the commercial ILP solver CPLEX using "callbacks" to implement the Gumbel randomization. This approach allows us to directly take advantage of CPLEX's optimized search heuristics (line 8 and 17 in Algorithm 1) and parallel solving capabilities, where multiple threads are used to explore the search tree and analyze open sub-problems in L in parallel. We experimented with up to 48 threads and observed significantly improved runtimes with more threads.

Exact Sampling

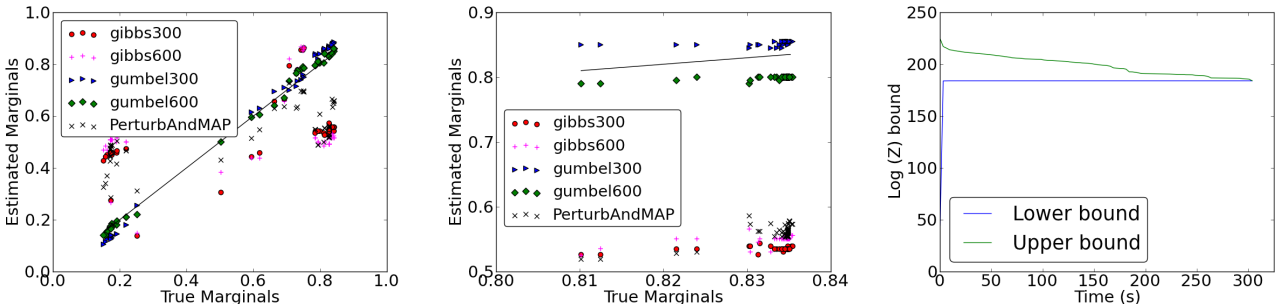
GumbelMIP, like all known exact sampling algorithms, needs exponential time in the worst case. In practice, however, models typically do not exhibit worst case behavior and modern ILP solvers employ a host of heuristics to speed up search. We evaluate the practical scalability of our approach.

We consider synthetic Ising models with n binary variables $x_i \in \{-1, 1\}$ and potentials $\psi_{ij}(x_i, x_j) = \exp(w_{ij}x_ix_j + f_ix_i + f_jx_j)$. Models with *mixed* interactions have i.i.d. w_{ij} drawn uniformly from $[-w, w]$, while *attractive* models have i.i.d. w_{ij} drawn uniformly from $[0, w]$. f_i are drawn uniformly from $[-1, 1]$.

In Figure 1a we report the runtime of GumbelMIP for randomly generated attractive clique-structured Ising models as a function of n. These models have treewidth n, making exact inference and sampling impractical for large n. In practice, exact state-of-the-art techniques run out of memory for $n > 35$. In contrast, our approach provides exact samples for models with up to $n = 60$. While the runtime appears to grow exponentially, the rate is milder and the memory usage limited. Further, we can exploit parallelism to drastically speed up the search even for a single sample (note the plot is in log-scale). While recently introduced hashing-based techniques can scale to models of similar size (Ermon et al. 2013), their results are approximate, while we provide exact samples. To the best of our knowledge, no existing technique can provide exact samples for models of this size.

To evaluate the effectiveness of the LP relaxations we use, we compare with an **A*-like sampling** algorithm that uses a simpler bounding method (Line 8 of the pseudocode). Specifically, we compute an upper bound on (4) as follows. For each term in the sum (i.e., for each potential), we find the largest value that is consistent with the current partial assignment to the ILP variables, and sum up these upper bounds. This relaxation is much weaker than an LP relaxation as it does not enforce consistency of variable values across factors, but can be evaluated more efficiently. For ease of implementation, we still employ perturbation inheritance. Figure 1a shows that GumbelMIP vastly outperforms this algorithm, which cannot produce any sample when $n > 35$ (with a 4 hour timeout). The use of powerful LP relaxations is thus crucial for the technique on high dimensional problems.

Next, we compare the runtime of GumbelMIP with a variant of **adaptive rejection sampling** scheme that can leverage combinatorial optimization and is therefore closest to GumbelMIP. Rejection sampling (Andrieu et al. 2003) relies on a proposal distribution that is tractable (easy to sample from) and upper bounds the desired target density. The closer these distributions are, the more efficient the sampling is. Obtaining a good proposal distribution typically requires domain-specific knowledge, but general proposal distributions can be constructed using ideas from optimization, as in the OS* algorithm (Dymetman, Bouchard, and Carter 2012). Specifically, we use a piecewise constant proposal distribution, obtained by partitioning the state space into J subtrees and computing an upper bound for $w(\sigma)$ in each of these J subtrees. We use an LP relaxation for this upper bound, a more sophisticated bounding technique than the ones in the original OS*. While any partition can in principle be used,



(a) Marginals on 8×8 grid. (b) Marginals on 8×8 grid. (c) Any-time bounds on $\log Z$.

Figure 2: Correlation vs true marginals after 5 and 10 minutes; any-time bounds on $\log Z$.

we obtain it following the branching heuristics employed by CPLEX when used to optimize w. We considered computing the partition and the corresponding upper bounds as a preprocessing step and we therefore did not include the time required to perform these steps in the runtime reported for rejection sampling. We see in Figure 1a (black line) that rejection sampling is faster than GumbelMIP for smaller instances, but the runtime grows very quickly and it cannot solve instances beyond $n = 35$.

Finally, to test our hypothesis that "simpler" models might be easier to solve, we consider **tree-structured graphical models**. These are tractable and support exact inference (and sampling). GumbelMIP is general-purpose and a priori oblivious to this special structure, but might be able to leverage it anyways because, e.g., the corresponding LP relaxations are known to be tight (Sontag et al. 2008). Indeed, we see in Figure 1b that scalability is drastically improved with respect to Figure 1a. Similar results are also obtained on other tractable classes, e.g., fully disconnected models.

Approximate Sampling and Any-time Bounds on Z

GumbelMIP can be used as an approximate method when the (perturbed) optimization problem is not solved to optimality, with formal guarantees provided earlier.

To evaluate the empirical effectiveness of this approach, we consider grid structured Ising models with low treewidth for which we can compute ground truth marginal probabilities, and evaluate the quality of the samples obtained if the search is aborted before an optimal solution is found. In Figures 2a (attractive case) and 2b (mixed interactions) we compare with ground truth (using a scatter plot) the marginals obtained by running GumbelMIP and a Gibbs sampler each for 5 and 10 minutes respectively (using 200 independent samples). We see that in both cases GumbelMIP provides more accurate marginals than the Gibbs sampler. The powerful heuristics employed by our optimizer are able to quickly find a near-optimal solution, even though the solver cannot prove its optimality. GumbelMIP therefore makes better use of the limited computational resources available. We also compare with the PerturbAndMap approximate sampler (Hazan, Maji, and Jaakkola 2013) based on the use of low dimensional Gumbel perturbations and exact optimization. Although PerturbAndMap is faster, the samples it generates can be quite inaccurate, emphasizing the need for exact, high-dimensional Gumbel perturbations used by GumbelMIP.

Figure 2c reports the any-time upper and lower bounds on the partition function using Theorem 2. We see that CPLEX finds an optimal solution very quickly (steep blue curve), spending most of the runtime proving its optimality.

Figure 3: Samples from a blocked Gibbs sampler after 100 iterations, for various block sizes.

Blocked Gibbs with Large Blocks

GumbelMIP can be used to sample from arbitrary probability distributions. In particular, it can be used to sample from posterior distributions over relatively large subsets of variables (beyond what could be done by brute force). It can therefore be used as a black box inside a blocked Gibbs sampler. Rather than solving a single (perturbed) optimization problem over all the variables, one can solve a sequence of smaller sub-problems over subsets of the variables (the block), while keeping the other variables fixed. This resembles a block coordinate ascent approach, except the randomness is resampled after each coordinate ascent step. Since GumbelMIP produces exact samples, the asymptotic properties of the blocked Gibbs sampler are preserved. The key advantage is that by using large block sizes one can drastically reduce mixing times. In the extreme case where the block includes all the variables, the chain mixes in one step.

We demonstrate the effectiveness of this approach on a Restricted Boltzmann Machine (RBM) with $v = 196$ visible units and $h = 100$ hidden units, trained on MNIST (handwritten digits) with constrastive divergence (Carreira-Perpinan and Hinton 2005). We use a simpler and more ef-

fective ILP encoding for RBMs, replacing 4 binary variable per pairwise factor with just one continuous variable tied to the corresponding pairwise potential. In every iteration of our blocked Gibbs sampler, we randomly select a block of k variables and sample from their posterior, given all the other variables. The case $k = 1$ corresponds to a traditional Gibbs sampler with random ordering of variable updates. In Figure 3 we plot the state of the chain every 100 iterations, for various block sizes. It is clear that using large block sizes makes the chain converge significantly faster, although the cost per iteration increases. Understanding the tradeoffs involved is largely empirical and a subject for future research.

Conclusions

We introduced GumbelMIP, a novel exact sampler for discrete probability distributions. By translating a sampling problem to a (perturbed) optimization problem, we can leverage a host of techniques from combinatorial optimization and operations research, including LP relaxations, branching heuristics, and parallel multi-threaded search. The resulting algorithm is very efficient and the first one to provide exact samples for high-treewidth models. Even when the resulting problems cannot be solved to optimality, approximate solutions provide high quality samples, as explained by our novel analysis of suboptimal solutions and tools to rigorously characterize the quality of the samples in terms of LP relaxations. Our technique can be used inside a blocked Gibbs sampler, allowing a tradeoff between more iterations and exact samples from larger blocks.

Acknowledgments

This work was supported by the Future of Life Institute (grant 2015-143902) and by the National Science Foundation Graduate Research Fellowship (grant DGE-114747).

References

Andrieu, C.; de Freitas, N.; Doucet, A.; and Jordan, M. I. 2003. An introduction to MCMC for machine learning. *Machine learning* 50(1-2):5–43.

Boehning, R. L.; Butler, R. M.; and Gillett, B. E. 1988. A parallel Integer Linear Programming algorithm. *European Journal of Operational Research* 34(3):393–398.

Carreira-Perpinan, M., and Hinton, G. 2005. On contrastive divergence learning. In *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 17.

Chakraborty, S.; Fremont, D. J.; Meel, K. S.; Seshia, S. A.; and Vardi, M. Y. 2014. Distribution-aware sampling and weighted model counting for SAT. In *Proc. of the 28th National Conference on Artifical Intelligence (AAAI)*, 1722–1730.

Dymetman, M.; Bouchard, G.; and Carter, S. 2012. The OS* algorithm: a joint approach to exact optimization and sampling. *CoRR* abs/1207.0742.

Ermon, S.; Gomes, C. P.; Sabharwal, A.; and Selman, B. 2013. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proc. of the 30th International Conference on Machine Learning (ICML)*.

Gane, A.; Hazan, T.; and Jaakkola, T. 2014. Learning with maximum a-posteriori perturbation models. In *Proc. of the 17th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*.

Gogate, V., and Dechter, R. 2007. Approximate counting by sampling the backtrack-free search space. In *Proc. of the 22nd National Conference on Artifical Intelligence (AAAI)*, volume 22, 198–203.

Gogate, V., and Dechter, R. 2011. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.

Gumbel, E. J., and Lieblein, J. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office Washington.

Hazan, T., and Jaakkola, T. 2012. On the partition function and random maximum a-posteriori perturbations. In *Proc. of the 29th International Conference on Machine Learning (ICML)*.

Hazan, T.; Maji, S.; and Jaakkola, T. 2013. On sampling from the gibbs distribution with random maximum a-posteriori perturbations. In *Advances in Neural Information Processing Systems*, 1268–1276.

Jerrum, M., and Sinclair, A. 1997. The markov chain monte carlo method: An approach to approximate counting and integration. In *Approximation Algorithms for NP-hard Problems*. Boston, MA: PWS Publishing. 482–520.

Johnson, E. L.; Nemhauser, G. L.; and Savelsbergh, M. W. 2000. Progress in Linear Programming-based algorithms for Integer Programming: An exposition. *INFORMS Journal on Computing* 12(1):2–23.

Kappes, J. H.; Swoboda, P.; Savchynskyy, B.; Hazan, T.; and Schnörr, C. 2015. Probabilistic correlation clustering and image partitioning using perturbed multicuts. In *Scale Space and Variational Methods in Computer Vision*. Springer. 231–242.

Kim, C.; Sabharwal, A.; and Ermon, S. 2015. Exact sampling with integer linear programs and random perturbations. Technical report, Stanford University.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT Press.

Leadbetter, R.; Lindgren, G.; and Rootzén, H. 1983. *Extremes and related properties of random sequences and processes*. Springer series in statistics. Springer-Verlag.

Maddison, C. J.; Tarlow, D.; and Minka, T. 2014. A* sampling. In *Advances in Neural Information Processing Systems*, 3086–3094.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Papandreou, G., and Yuille, A. L. 2011. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 193–200. IEEE.

Schlesinger, D. 2009. General search algorithms for energy minimization problems. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 84–97. Springer.

Sontag, D.; Meltzer, T.; Globerson, A.; Jaakkola, T.; and Weiss, Y. 2008. Tightening LP relaxations for MAP using message passing. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 503–510.

Tarlow, D.; Adams, R. P.; and Zemel, R. S. 2012. Randomized optimum models for structured prediction. *Journal of Machine Learning Research - Workshop and Conference Proceedings* 22:1221–1229.

Wainwright, M. J., and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1(1-2):1–305.

Wolsey, L. A., and Nemhauser, G. L. 2014. *Integer and combinatorial optimization*. John Wiley & Sons.

Appendix: Proofs

Proof of Theorem 1. Let Y_i be the event that $\forall j > i : w'(\sigma^{(i)}) \geq w'(\sigma^{(j)})$. From known properties of Gumbel perturbations, $\Pr_\gamma[Y_i] = \frac{w(\sigma^{(i)})}{\sum_{j \geq i} w(\sigma^{(j)})} = \frac{w(\sigma^{(i)})}{Z - \sum_{j<i} w(\sigma^{(j)})}$. We can rewrite Y_1 as the event that $w'(\sigma^{(1)}) \geq \max_{i \geq 2}\{w'(\sigma^{(i)})\}$. The event Y_2, which is equal to the event that $2 = \arg\max_{i \geq 2}\{w'(\sigma^{(i)})\}$, is independent of both the values of $w'(\sigma^{(1)})$ (because the γ_σ's are independent) and $\max_{i \geq 2}\{w'(\sigma^{(i)})\}$ (because the argmax and max of a set of independent Gumbel variables are independent). This implies that Y_1 is independent of Y_2. Therefore, $\Pr_\gamma[Y_1, Y_2] = \Pr_\gamma[Y_1]\Pr_\gamma[Y_2]$. Extending this argument, it may be verified that $\Pr_\gamma[Y_1, Y_2, \ldots, Y_k] = \prod_{i \leq k} \Pr_\gamma[Y_i]$, proving the claim. □

Proof of Proposition 1. σ^*, by design, has rank one among the configurations already explored by the algorithm. For each sub-problem (Q, X, γ_Q, σ) that remains in L, $w'_Q = \mathit{LPrelax}(Q) + \gamma_Q$ is an upper bound on $w'(\sigma)$ for all of the $2^{|X|}$ configurations σ in the feasible set of Q. Clearly, if $w'_Q < w'(\sigma^*)$, no configuration in Q will be ahead of σ^* in the ordering. Otherwise, in the worst case, all $2^{|X|}$ configurations in Q have weight $\mathit{LPrelax}(Q)$. In this case, configuration σ will, by design, surely be ahead of σ^*. Further, any of the remaining $2^{|X|} - 1$ configurations σ in Q will be ahead of σ^* if $w'(\sigma^*) < w'(\sigma) \leq \mathit{LPrelax}(Q) + \gamma_\sigma$; thus, this happens with a probability bounded above by $\Pr_\gamma[\gamma_\sigma > w'(\sigma^*) - \mathit{LPrelax}(Q)]$ which is $1 - \Pr_\gamma[\gamma_\sigma \leq w'(\sigma^*) - \mathit{LPrelax}(Q)]$. Since γ_σ is distributed as $\mathrm{TruncGumbel}(0, \gamma_Q)$, the last probability expression is simply the CDF of $\mathrm{TruncGumbel}(0, \gamma_Q)$ evaluated at $w'(\sigma^*) - \mathit{LPrelax}(Q)$. Using linearity of expectation across the $2^{|X|} - 1$ configurations in S that behave this way, and adding one for σ, we obtain an upper bound on the expected number of configurations in S ahead of σ^* in the ranking. Summing over all such Q gives the desired result. □

Proof of Theorem 2. For the bounds on the expectation, fix any γ and consider any stopping point. v^* is the evaluation of the objective function $\log w(\sigma) + \gamma_\sigma$ for some σ, and is therefore a lower bound on the optimal value. If the optimal σ^* corresponds to a search node we have already explored, then $\log w(\sigma^*) + \gamma_{\sigma^*} = v^*$. Otherwise, for the subproblem Q whose feasible set contains σ^*, we have $\log w(\sigma^*) \leq \mathit{LPrelax}(Q)$ and $\gamma_{\sigma^*} \leq \gamma_Q$. Therefore,

$$v^* \leq \max_\sigma \log w(\sigma) + \gamma_\sigma \leq \max_{(Q, \gamma_Q, k) \in L} \mathit{LPrelax}(Q) + \gamma_Q.$$

Taking expectations over γ, we obtain

$$\mathbb{E}_\gamma[v^*] \leq \mathbb{E}_\gamma[\max_\sigma \log w(\sigma) + \gamma_\sigma] \leq \mathbb{E}_\gamma[U].$$

The middle expression here, from eq. (2), is precisely $\log Z + C_E$, proving the first part of the theorem.

We next show sample averages over sufficiently many runs can be used to closely approximate $\mathbb{E}_\gamma[v^*]$ and $\mathbb{E}_\gamma[U]$, and can thus provide high probability lower and upper bounds on $\log Z + C_E$. In the t-th iteration, let $w'^*_t = \max_\sigma \log w(\sigma) + \gamma^{(t)}_\sigma$. Then w'^*_t is distributed with mean $\log Z + C_E$ and variance $\pi^2/6$. From the above argument, we know that $v^*_t \leq w'^*_t$. Using Cantelli's inequality,

$$\begin{aligned}
&\Pr\left[\frac{1}{T}\sum_{t=1}^{T} v^*_t < (\log Z + C_E) + \epsilon\right] \\
&\geq \Pr\left[\frac{1}{T}\sum_{t=1}^{T} w'^*_t < (\log Z + C_E) + \epsilon\right] \\
&= 1 - \Pr\left[\frac{1}{T}\sum_{t=1}^{T} w'^*_t \leq (\log Z + C_E) + \epsilon\right] \\
&\geq 1 - \frac{\pi^2/(6T)}{\epsilon^2 + \pi^2/(6T)} \\
&= 1 - \frac{1}{6T(\epsilon/\pi)^2 + 1}
\end{aligned}$$

Analogously,

$$\Pr\left[\frac{1}{T}\sum_{t=1}^{T} U_t > (\log Z + C_E) - \epsilon\right] \geq 1 - \frac{1}{6T(\epsilon/\pi)^2 + 1}$$

Setting $T = (1/\delta - 1)\pi^2/(6\epsilon^2)$ makes the bound become $1 - \delta$. Rearranging the terms then yields the desired result. □